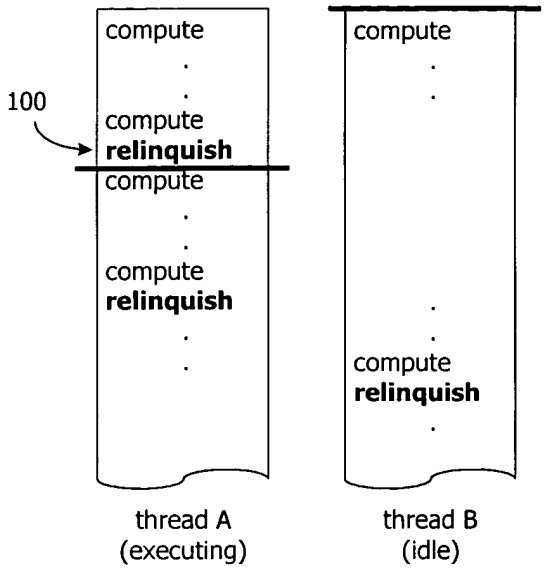
compute
100
compute
relinquish
compute
compute
relinquish
compute
relinquish
thread A
(executing)
thread B
(idle)

FIG. 1A

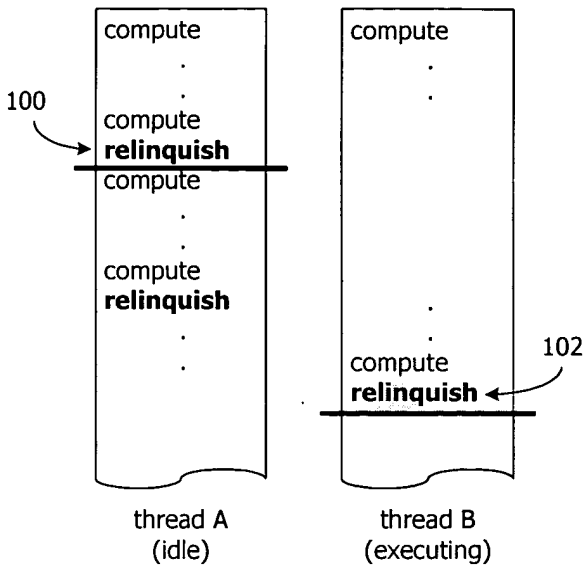
compute
100
compute
relinquish
compute
compute
relinquish
compute
relinquish
102
thread A
(idle)
thread B
(executing)

FIG. 1B

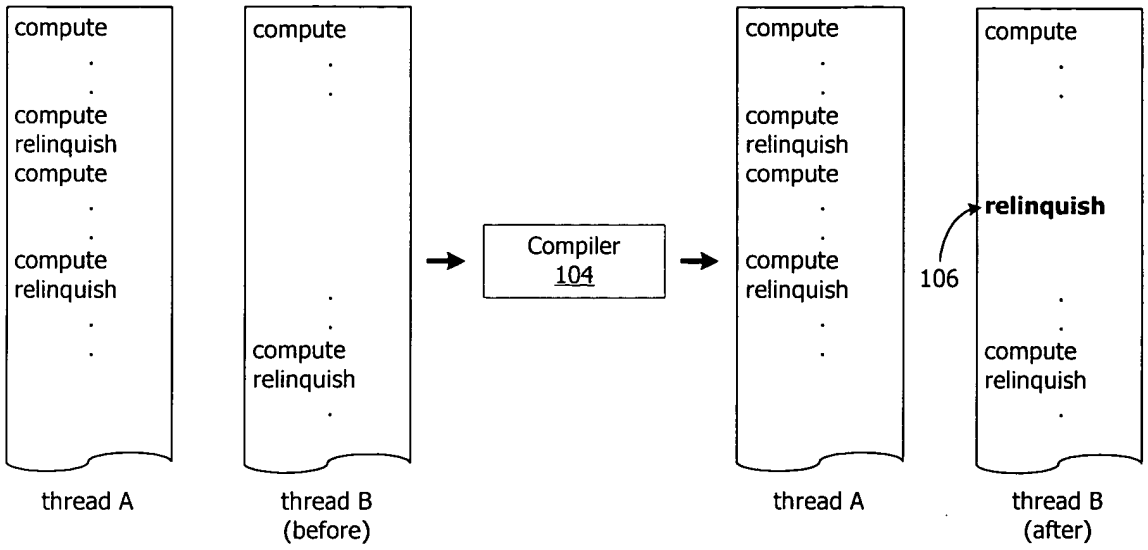
compute
compute
relinquish
compute
compute
relinquish
thread A
compute
compute
relinquish
thread B
(before)
Compiler
104
compute
compute
relinquish
compute
compute
relinquish
thread A
106
compute
relinquish
compute
relinquish
thread B
(after)

FIG. 2

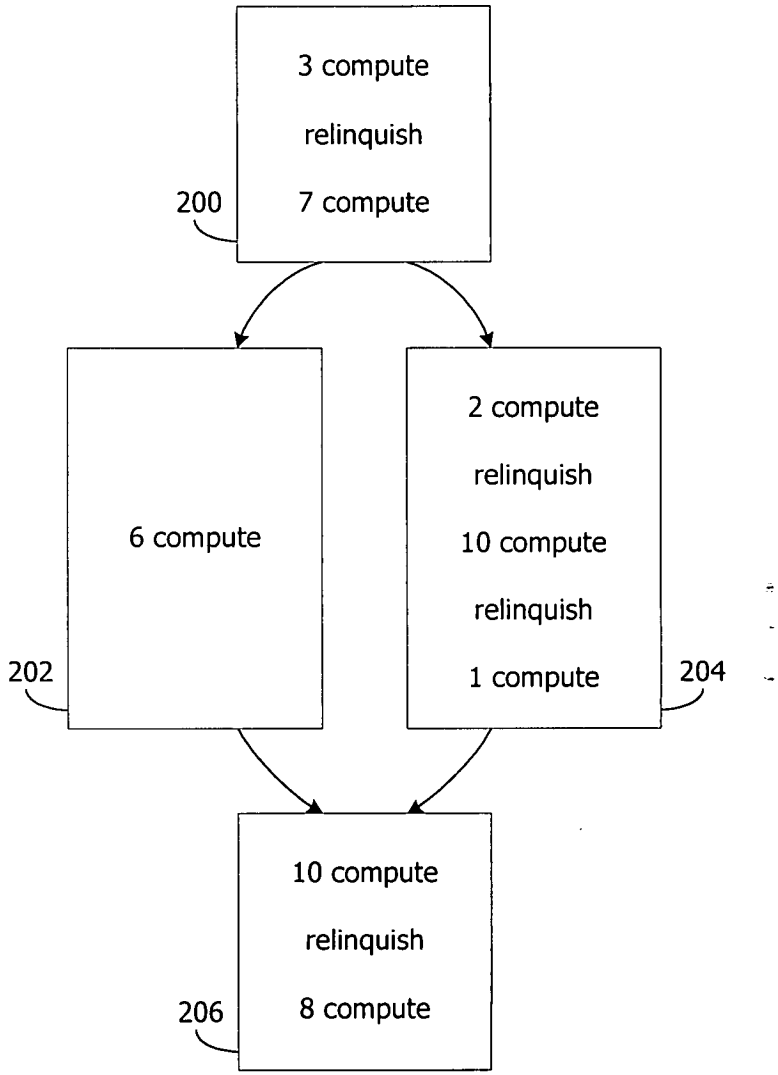
3 compute
relinquish
7 compute
200
6 compute
202
2 compute
relinquish
10 compute
relinquish
1 compute
204
10 compute
relinquish
8 compute
206

FIG. 3A

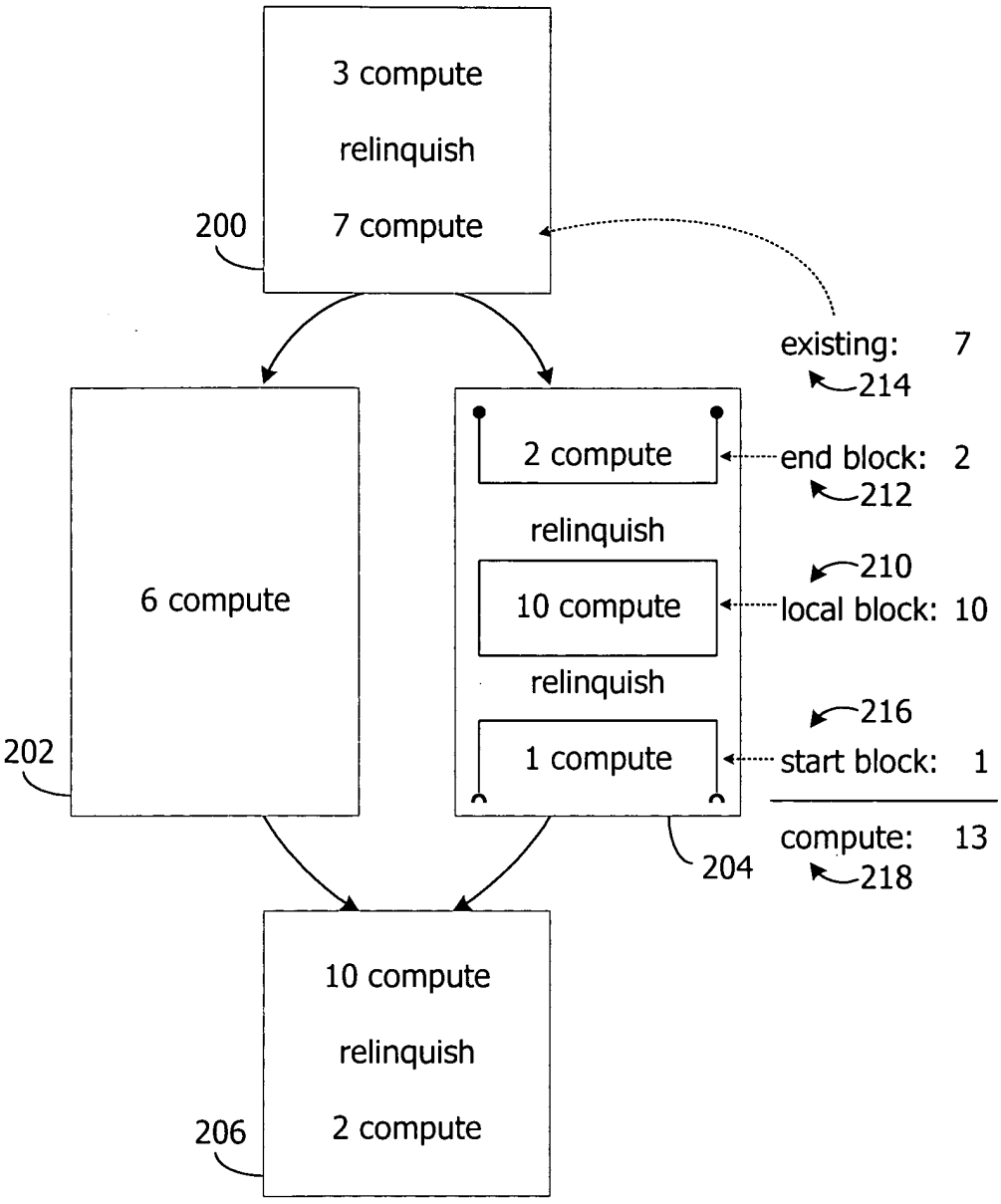
3 compute
relinquish
7 compute
200
existing: 7
214
6 compute
202
2 compute
end block: 2
212
relinquish
210
10 compute
local block: 10
relinquish
216
1 compute
start block: 1
compute: 13
204
218
10 compute
relinquish
2 compute
206

FIG. 3B

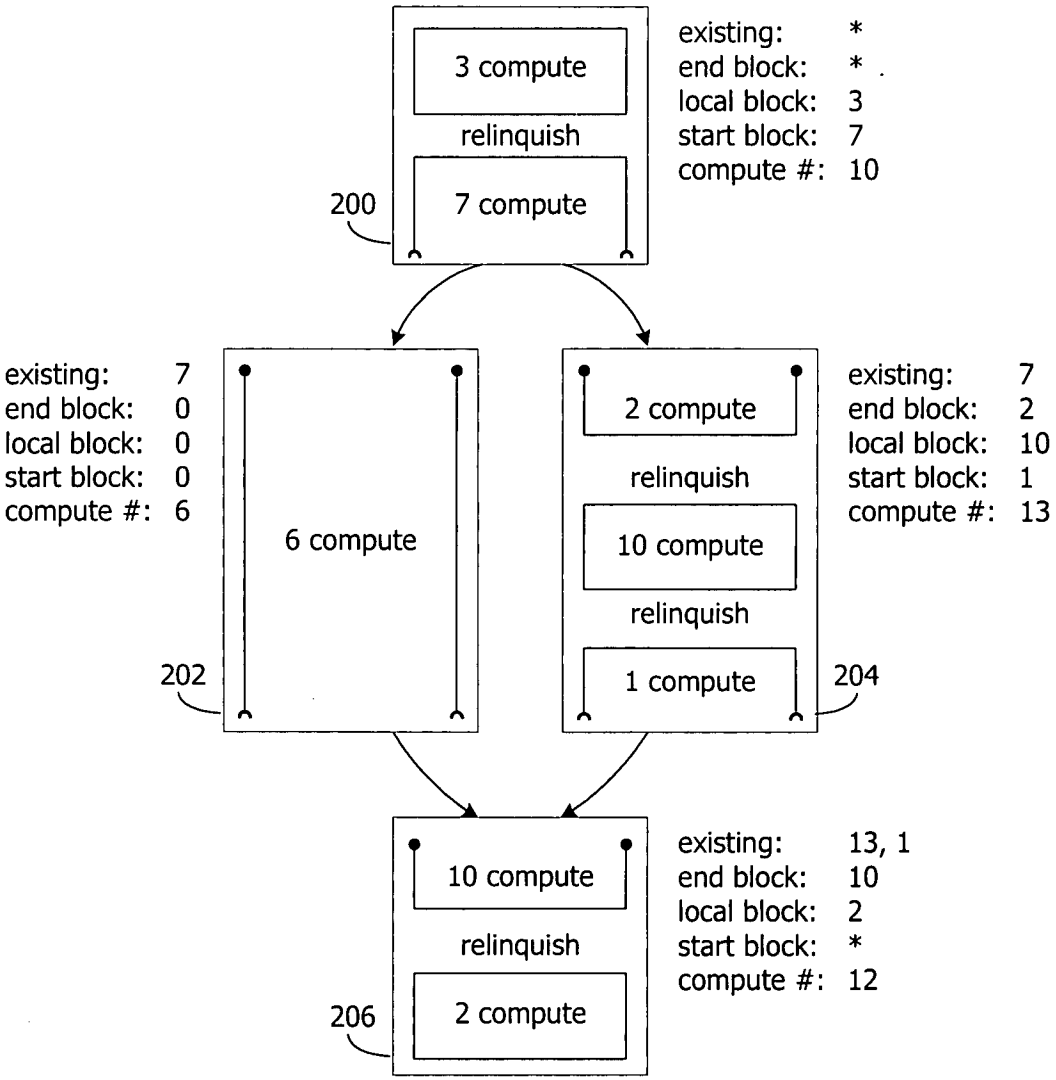
3 compute
relinquish
7 compute
200
existing: *
end block: *
local block: 3
start block: 7
compute #: 10
existing: 7
end block: 0
local block: 0
start block: 0
compute #: 6
6 compute
202
2 compute
relinquish
10 compute
relinquish
1 compute
204
existing: 7
end block: 2
local block: 10
start block: 1
compute #: 13
10 compute
relinquish
2 compute
206
existing: 13, 1
end block: 10
local block: 2
start block: *
compute #: 12

FIG. 3C

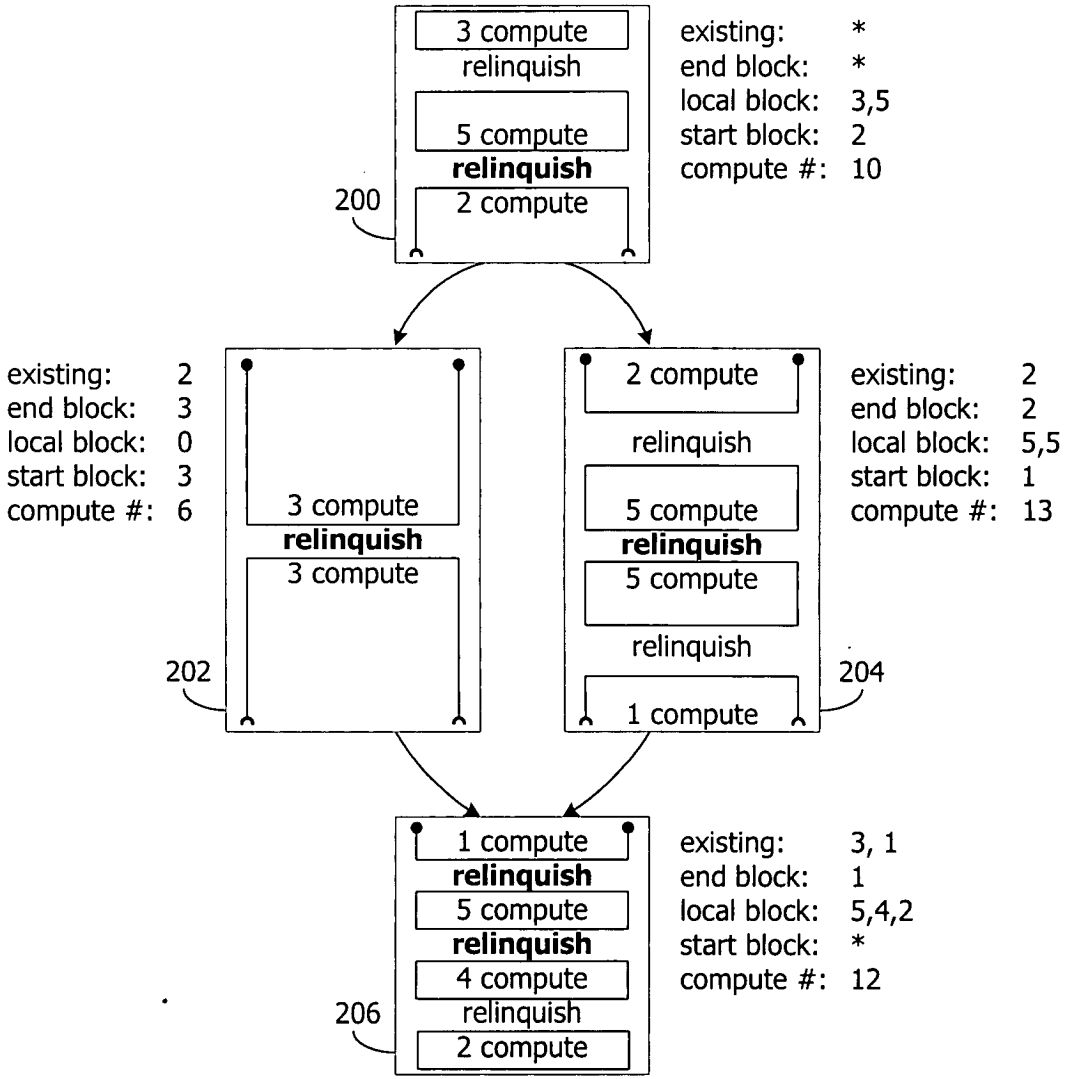
3 compute
relinquish
5 compute
relinquish
2 compute
200
existing: *
end block: *
local block: 3,5
start block: 2
compute #: 10
existing: 2
end block: 3
local block: 0
start block: 3
compute #: 6
3 compute
relinquish
3 compute
202
2 compute
relinquish
5 compute
relinquish
5 compute
relinquish
1 compute
204
existing: 2
end block: 2
local block: 5,5
start block: 1
compute #: 13
1 compute
relinquish
5 compute
relinquish
4 compute
relinquish
2 compute
206
existing: 3, 1
end block: 1
local block: 5,4,2
start block: *
compute #: 12

FIG. 3D

```
// for wholly included compute blocks
for each compute block wholly contained in node
        if block_size > threshold
300
                number_blocks = ceiling(block_size, threshold)
                insert relinquish instructions to break up block into ~ equal
                number_blocks
```

FIG. 4A

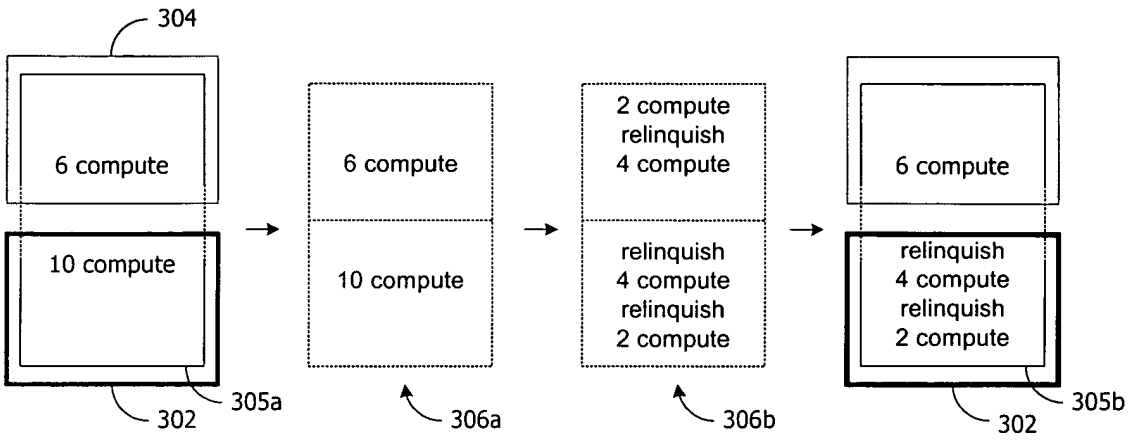
304
6 compute
10 compute
302
305a
6 compute
10 compute
306a
2 compute
relinquish
4 compute
relinquish
4 compute
relinquish
2 compute
306b
6 compute
relinquish
4 compute
relinquish
2 compute
302
305b

FIG. 4B

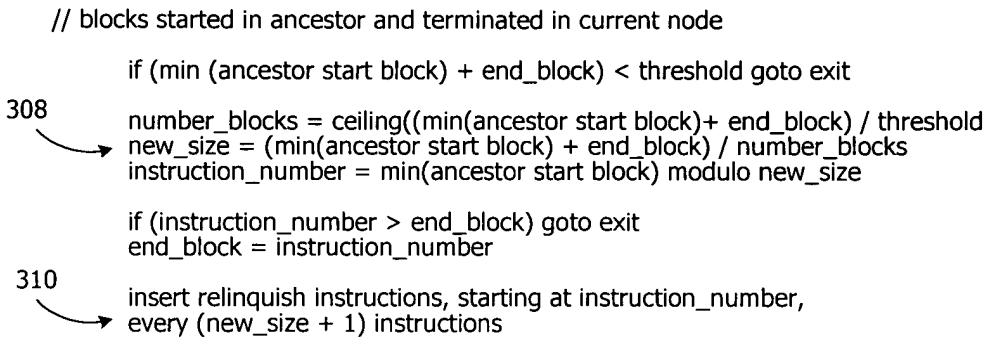
// blocks started in ancestor and terminated in current node
if (min (ancestor start block) + end_block) < threshold goto exit
308
number_blocks = ceiling((min(ancestor start block)+ end_block) / threshold
new_size = (min(ancestor start block) + end_block) / number_blocks
instruction_number = min(ancestor start block) modulo new_size
if (instruction_number > end_block) goto exit
end_block = instruction_number
310
insert relinquish instructions, starting at instruction_number,
every (new_size + 1) instructions

FIG. 4C

```
// blocks started in this node and terminated in descendent
if (start_block + min (descendent end block)) < threshold skip this processing
// Determine where to insert the first relinquish instruction
312 ⟶ number_blocks = ceiling((start_block + min(descendent end block) / threshold)
      new_size = (start_block + min (descendent end block) / number_blocks
      instruction_number = min (descendent end block) modulo new_size
314 ⟶ insert relinquish instructions, starting instruction_number
      from the end of the node, every (new_size + 1) instructions
```

FIG. 4D

```
// for nodes having no relinquish instructions

316  // Determine size of smallest contiguous block of instructions
     size = min ( descendent end block) + compute_count + min (ancestor start block)
     if size < threshold goto exit

     // Compute where to insert first relinquish instruction in this node
     number_blocks = ceiling(size / threshold)
     new_size = size / number_blocks
318  instruction_number = min (ancestor start block) modulo new_size
     if instruction_number > compute_count goto exit

     insert relinquish instructions, starting at
     instruction_number, every (new_size + 1) instructions
```

FIG. 4E

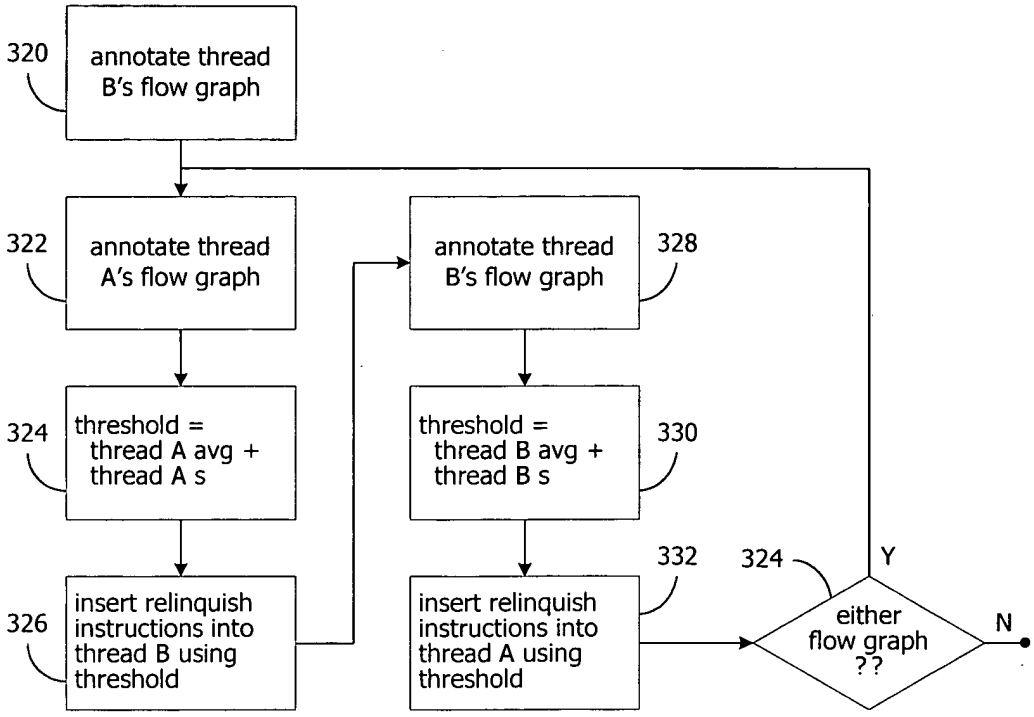
320
annotate thread B's flow graph
322
annotate thread A's flow graph
324
threshold = thread A avg + thread A s
326
insert relinquish instructions into thread B using threshold
328
annotate thread B's flow graph
330
threshold = thread B avg + thread B s
332
insert relinquish instructions into thread A using threshold
324
either flow graph ??
Y
N

FIG. 5

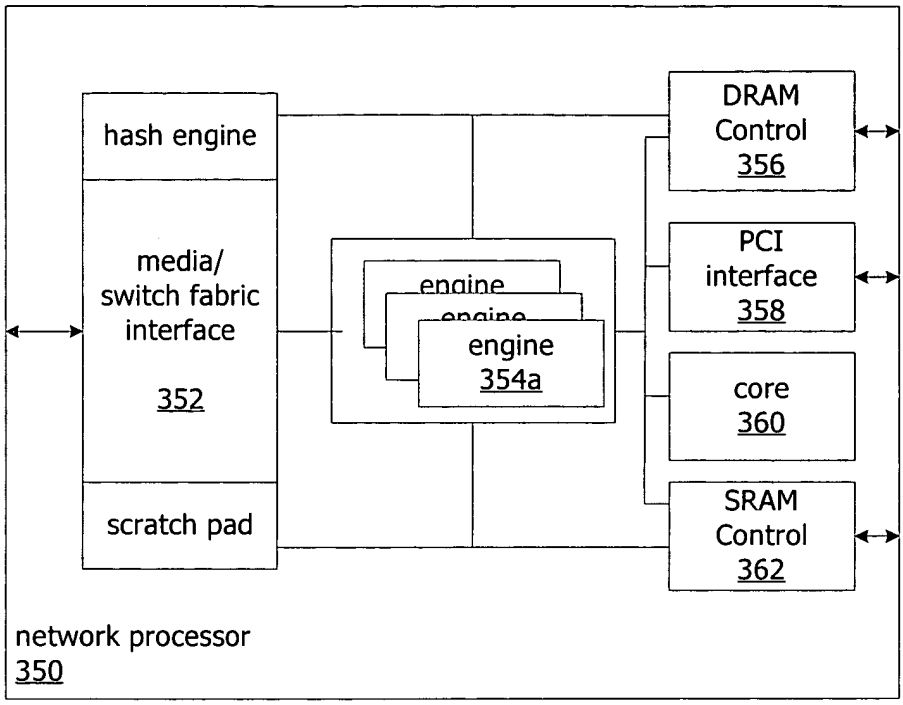
hash engine
media/
switch fabric
interface
352
scratch pad
engine
engine
engine
354a
DRAM
Control
356
PCI
interface
358
core
360
SRAM
Control
362
network processor
350

FIG. 6